Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

По дисциплине: «МиАПР»

Тема: «Нелинейные ИНС в задачах прогнозирования. Адаптивный шаг обучения»

Выполнил:

Студент 2 курса

Группы ПО-7

Рекун И.В.

Проверил:

Крощенко А.А.

2021

**Цель работы:**

изучить обучение и функционирование линейной ИНС при решении

задач прогнозирования.

**Задание:**

Написать на любом ЯВУ программу моделирования прогнозирующей нелинейной ИНС. Для тестирования использовать функцию

y = a\*cos(b\*x) + c\*sin(d\*x)

a=0.1, b=0.1, c=0.05, d=0.1, кол-во входов ИНС = 6, кол-во НЭ в скрытом слое = 2.

Для прогнозирования использовать многослойную ИНС с одним скрытым слоем. В качестве функций активации для скрытого слоя использовать сигмоидную функцию, для выходного - линейную.

**Код программы:**

from math import sin, cos, exp

import random

input\_neuron = 6

hidden\_neuron = 2

min\_error = 1e-9

alpha = 0.1

step = 0.1

weight\_i\_j = [

[random.uniform(-1, 1) for \_ in range(hidden\_neuron)]

for \_ in range(input\_neuron - 1)

]

weight\_i\_k = [random.uniform(-1, 1) for \_ in range(hidden\_neuron)]

def func(x):

a, b, c, d = 0.2, 0.4, 0.09, 0.4

return a \* cos(b \* x \* step) + c \* sin(d \* x \* step)

def idl\_value(x):

y = []

for i in range(input\_neuron):

y.append(func(x))

return y

def calculate\_hidden(j, y, T\_hidden):

S\_hidden = 0

for i in range(input\_neuron - 1):

S\_hidden += y[i] \* weight\_i\_j[i][j]

S\_hidden -= T\_hidden[j]

return S\_hidden

def calculate\_out(y\_hidden, T\_out):

S\_out = 0

for i in range(hidden\_neuron):

S\_out += y\_hidden[i] \* weight\_i\_k[i]

S\_out -= T\_out

return S\_out

def sigmoid(j, y, T\_hidden):

S\_hidden = calculate\_hidden(j, y, T\_hidden)

return 1 / (1 + exp(-S\_hidden))

def Wjk\_change(y\_hidden, error,step\_k):

for i in range(hidden\_neuron):

weight\_i\_k[i] -= step\_k \* error \* y\_hidden[i]

def Wij\_change(y\_hidden, error, y,step\_j):

for i in range(input\_neuron - 1):

for j in range(hidden\_neuron):

weight\_i\_j[i][j] -= (

step\_j

\* error

\* weight\_i\_k[j]

\* y[i]

\* (y\_hidden[j] \* (1 - y\_hidden[j]))

)

def calculate\_T\_hidden(T\_hidden, y\_hidden, error, step\_hidden):

for i in range(hidden\_neuron):

T\_hidden[i] += step\_hidden \* error \* y\_hidden[i] \* (1 - y\_hidden[i])

def calculate\_T\_out(T\_out, local\_error, step\_out):

T\_out += local\_error \* step\_out

def adaptive\_step(y\_hidden, y\_out, error):

num, comp\_1 = 1, 1

error\_j = []

for i in range(hidden\_neuron):

error\_j.append(error \* y\_out \* (1 - y\_out) \* weight\_i\_k[i])

for i in range(hidden\_neuron):

num += (error\_j[i] \*\* 2) \* y\_hidden[i] \* (1 - y\_hidden[i])

comp\_1 += (error\_j[i] \*\* 2) \* (y\_hidden[i] \*\* 2) \* ((1 - y\_hidden[i]) \*\* 2)

step\_j = (4 \* num) / ((1 + y\_out \*\* 2) \* comp\_1)

return step\_j

def main():

epoch = 0

sum\_error = 1

T\_hidden = [0 for \_ in range(hidden\_neuron)]

T\_out = 0

step\_hidden = 0

step\_out = 0

while sum\_error > min\_error:

for k in range(30):

y\_hidden = []

y\_out, sum\_error = 1, 0

y\_ideal = idl\_value(k)

for i in range(hidden\_neuron):

y\_hidden.append(sigmoid(i, y\_ideal, T\_hidden))

y\_out = calculate\_out(y\_hidden, T\_out)

local\_error = y\_out - y\_ideal[0]

step\_hidden = adaptive\_step(y\_hidden, y\_out, local\_error)

step\_out = 1 / (1 + (y\_hidden[0] \*\* 2) + (y\_hidden[1] \*\* 2))

Wjk\_change(y\_hidden, local\_error,step\_out)

Wij\_change(y\_hidden, local\_error, y\_ideal,step\_hidden)

calculate\_T\_hidden(T\_hidden, y\_hidden, local\_error,step\_hidden)

calculate\_T\_out(T\_out, local\_error,step\_out)

sum\_error += 0.5 \* (local\_error \*\* 2)

epoch += 1

print(f'Number epoch:{epoch}:{local\_error}')

print(f"Number of epoch: {epoch}")

print("Testing result:")

print(

"{:<30}{:<30}{}".format("Reference value", "Current value", "Error")

)

for i in range(30, 45):

y\_hidden\_test = []

y\_out\_test, sum\_error\_ = 0, 0

y\_ideal\_test = idl\_value(i)

for j in range(hidden\_neuron):

y\_hidden\_test.append(sigmoid(j, y\_ideal\_test, T\_hidden))

y\_out\_test = calculate\_out(y\_hidden\_test, T\_out)

local\_error\_ = y\_out\_test - y\_ideal\_test[-1]

sum\_error\_ += 0.5 \* (local\_error\_ \*\* 2)

calculate\_T\_hidden(T\_hidden, y\_hidden\_test, local\_error,step\_hidden)

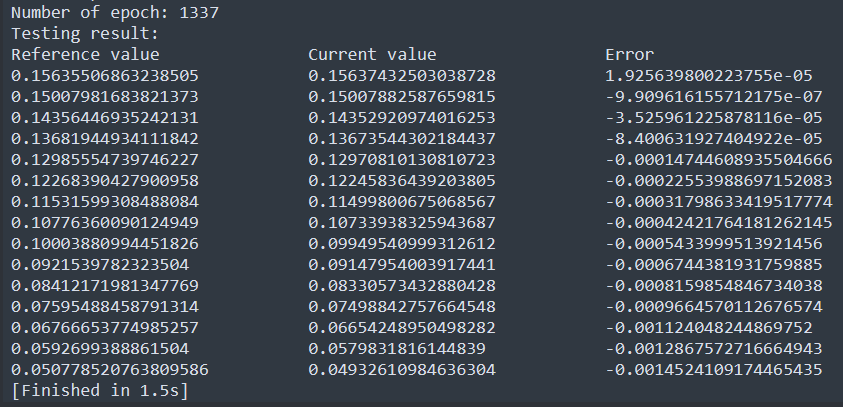
calculate\_T\_out(T\_out, local\_error,step\_out)

print(f"{y\_ideal\_test[-1]:<29} {y\_out\_test:<29} {local\_error\_}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Результат:**



**Вывод:**

Судя по результатам работы нейронная сеть с адаптивным шагом обучается быстрее, а также достигается меньшая ошибка.

Изучил обучение и функционирование многослойной ИНС при решении

задач прогнозирования на ЯП python.